Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Ivan Antunović

Izrada računalne igre u okruženju Phaser

Završni rad

Pula, rujan, 2019.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Ivan Antunović

Izrada računalne igre u okruženju Phaser

Završni rad

JMBAG: 0303068625 8,redoviti student

Studijski smjer: Informatika

Predmet: Napredne tehnike programiranja

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujan, 2019.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Ivan Antunović, kandidat za prvostupnika Informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, Ivan Antunović dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom 'Izrada računalne igre u okruženju Phaser' koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SAŽETAK

Računalna igra koja je ovdje prikazna napravljena je u okruženju Phaser 3 koje internira s HTML-om jer se prikazuje u pretraživaču uz pomoć 'node.js-a' .Namijenjena je prvenstveno za zabavu, iako bi mogla imati i primjenu kod psihotestova pri provjeri za zaposlenje/izdavanja raznih potvrda za koje je potreban imati dobar očni refleks ,brzinu reagiranja i brzih promišljanja. Ovakav način rada kod provjera koristi se npr. u regrutiranju američkih vojnih pilota,u školovanju kontrolora leta,raznih motritelja postrojenja itd. Igrač u ovoj igri pokušava imati što bolje rezultate, prepreke koja ga sputavaju su 'protivnički zrakoplovi' koji lete prema njemu,povećavaju svoju brzinu što je igrač bolji,a time i jači efekt iznenađenja pošto svaki zrakoplov ima drugačiji način kretanja i brzinu te igrač ne može niti u jednom trenutku biti siguran odakle će krenuti idući protivnik.

Ključne riječi:*Phaser3,računalna igra,node.js,HTML,očni refleks,pretraživač,igrač, efekt iznenađenja*

ABSTRACT

The computer game shown here is made in a Phaser 3 environment which interacts with HTML because it is rendered in the browser by 'node.js'. It is intended primarily for entertainment, although it could also be used for psychotesting when checking for employment / issuing various certificates for which it is needed to have a good eye reflex, responsiveness and quick reflections. This mode of check is used for example in recruiting US military pilots, training flight controllers, various plant monitors, etc. The player in game tries to get the best results, the obstacles that keep him down are 'enemy planes' flying towards him, increasing its speed the better the player, and thus the stronger the effect of surprise since each aircraft has a different mode of movement and speed, and the player can not at any moment be sure from where the next opponent will go.

Keywords*: Phaser3, computer game, node.js, HTML, ocular reflex, browser, player, the effect of surprise*

SADŽAJ

1. Uvod

Tema ovog završnog je razvijanje računalne igre pomoću knjižnice 'Phaser3' koja je napravljena u jeziku 'JavaScript'. Konačni proizvod se pokreće pomoću nekog od pretraživača poput 'Microsoft Edgea' zbog toga što se datoteke koje grade igru naposljetku prevode u HTML 5.0 jezik. Igra se sastoji od jednog igrača koji unutar igre posjeduje jedan 'borbeni zrakoplov' ,a prema njemu se iz različitih smjerova s različitim kretanjem kreću protivnici.

Cilj igre je izdržati s tri virtualna života što je više moguće i testiranje vlastitih refleksa, posebice dolazi do izražaja refleks oka i brzina reagiranja igrača. Igranje postaje sve kompliciranije što igrač ima bolji rezultat pošto brzina kretanja protivničkih zrakoplova ovisi o vrijednosti rezultata igrača. Svaki neprijateljski zrakoplov posjeduje brzinu koja je proporcionalna kroz igru s rezultatom igrača, ali se razlikuju po jednadžbi brzine i po smjeru kretanja. To znači da svaki put kada igrač pogodi neprijatelja, brzina ostalih se poveća, a kada igrač biva pogođen, ukoliko mu je ostalo još životnih bodova, brzina neprijatelja se naglo smanji te se isti resetiraju na početak pozadine.

Cijeli koncept je zamišljen kao koordinatni sustav 800x600 po kojem se izmjenjuju objekti (najčešće slike u formatu .png) i koji imaju međusobnu interakciju ukoliko dođu na isti dio (dijeljenje barem jedne točke) koordinatnog sustava. Rezultat i preostale šanse za nastavak igranja se učitavaju kao dio naslova na površini sustava u obliku 'labela'. Za kraj igranja se učitava konačni rezultat s porukom zahvale. Unutar igre su također dodani zvučni efekti i pozadinska glazba. Zvučni efekt eksplozije je namijenjen kao dodatni doživljaj kada igrač uništi jedan od protivničkih 'zrakoplova'.Za pravilan prikaz likova u igri i doživljaj zvuka potrebno je bilo koristiti i neke od online i Microsoft alata. Način na koji smo ih koristili i zbog čega je objašnjen po naslovom 'Alati i okruženje za izradu igre'.

Kontrole za igranju su sljedeće : strelice za kretanje i 'spacebar' za ispaljivanje 'projektila' po koordinatnom sustavu. Moguće se kretati i dijagonalno ukoliko koristimo jednu strelicu za lijevo/desno i jednu za gore/dole.Za potrebe igranja ove igre nije potrebna internetska veza, no potrebno je instaliranje node.js paketa.

2. Alati i okruženje korišteni za izradu igre

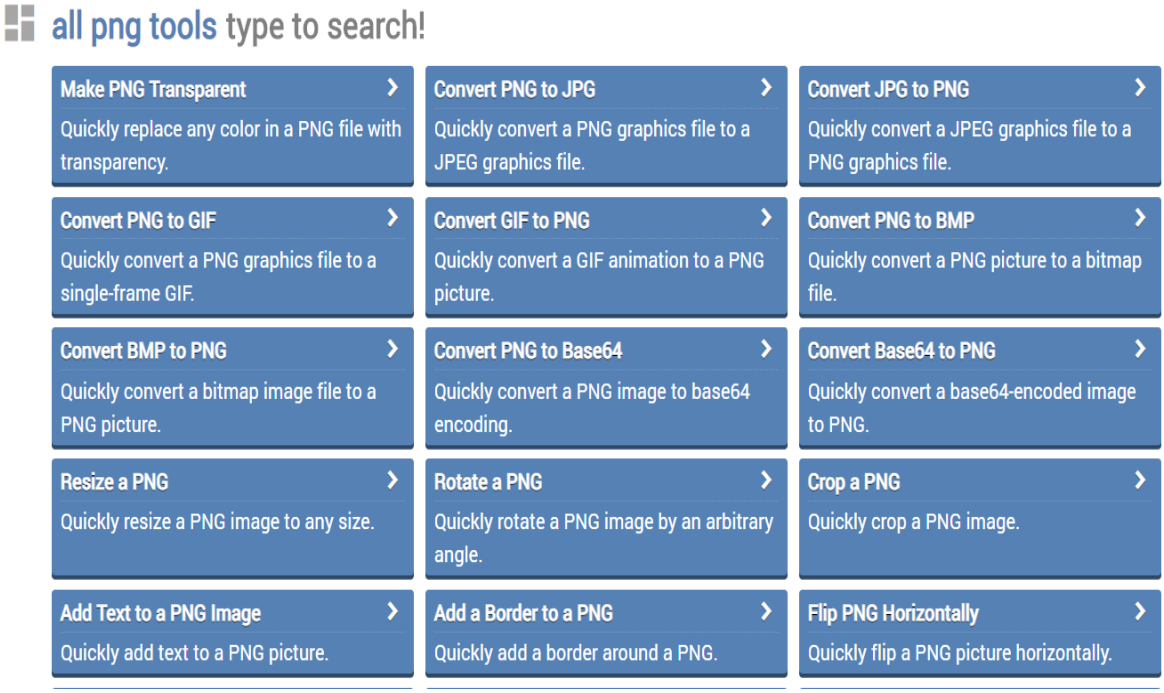
Za potrebu kreiranju ove igre korišteno je okruženje 'Phaser '(treće izdanje),node.js okruženje i alati za izradu slika i zvuka. Phaser je okruženje za izradu mobilnih i računalnih igara, namijenjenih za pretraživače koje podržavaju HTML5 canvas. Sastoji se od JavaScript biblioteke iz koje se kada se uključi u skriptu prvobitne datoteke 'Index.html' i uz pomoć paketa funkcija 'Pixi.js' koji služi za prijenos slika u obliku pixela daje završni oblik korisničkog sučelja u igri. Također posjeduje i svoj sistem fizike u virtualnom svijetu preko koordinatnog sustava,dok animaciju kreira kroz brzi pregled okvira jedne slike. U ovom projektu korišten je JavaScript iako se Phaser igra može implementirati i uz pomoć TypeScripta.

JavaScript kod se pokreće pomoću 'node.js' okruženja, sve datoteke, osim 'Index.html' su implementirane u JavaScriptu, te zbog toga imaju sufiks '.js'. Kako je već navedeno, 'Pixi.js' prevodi objekte u vizualni oblik dok funkcije promatraju naše karaktere u igru kao objekte koji posjeduju atribut pozicije,odnosno gdje se on nalazi u koordinatnom sustavu. Također interakcija između objekata definirana je kroz kod i uz pomoć takvog sustava sve izgleda kao da je u stvarnom vremenu te korisnik ima doživljaj kao se radi o stotinama objekata, a ne zapravo o sveukupnih njih deset.

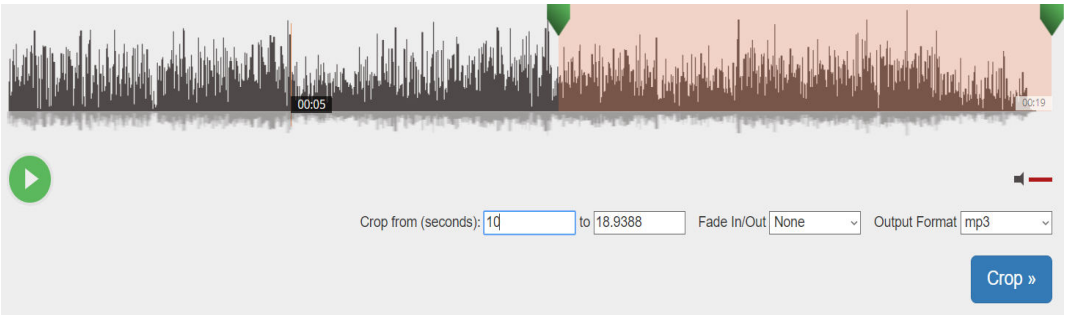
Za oblikovanje slika koje će služiti kao izgled naših objekata korišten je alat Microsofta 'Bojanje' i nekoliko online alata poput 'onlinePNGtools' (dio alata i opcija prikazuje slika 1.)koji osim što uređuju određenu sliku od nje mogu napraviti i ikonu koja je potrebna kako nam se ne bi cjelokupna pozadina slika prikazivala na našoj površini.

Osim vizualnog i funkcijskog djela, važno je napomenuti i zvuk. Kada bi odabrali melodiju, za potrebe igre morali bi određene dijelove formatirati ili srezati. Iako Phaser u sebi ima paket funkcija za uređivanje tj. konfiguriranje zvuka preko koda , određene dijelove smo morali uređivati uz pomoć i online alata 'audiotrimmer'(alat prikazan na slici 2.)

Linkove za online alate moguće je naći ispod slika u popisu literature.



Slika 1. Dio opcija pngonlinetoolsa [1]



Slika 2. Rad s audiotrimmerom [2]

3. Razrada funkcionalnosti

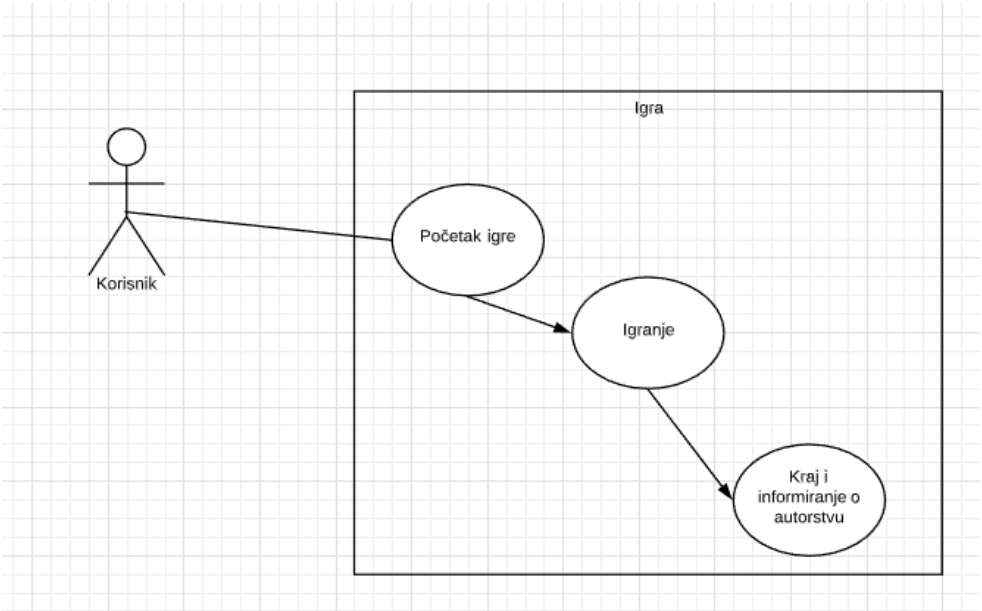
U ovom djelu opisano je razrada funkcionalnosti kroz dijagrame, spoj datoteka unutar jedne kao početne za pokretanje igre, objašnjavanje programskog koda,učitavanje audio-vizualnih atributa objekata i interakcija između objekata

Na sljedećoj slici vidimo dijagram aktivnosti unutar igre. Korisnik pri pokretanju igrice mora još jednom desnim klikom miša kliknuti da želi početi igru. To označava zadatak 'početak igre'. Nakon toga slijedi igranje čije trajanje ovisi o sposobnostima korisnika. Nakon izvjesnog vremena, svojevoljno ili prisilno od strane igre moraju završiti gdje dobiva informacije o autorstvu i zahvalu na igranju što je vidljivo iz dijagrama aktivnosti.

Funkcionalnosti igre:

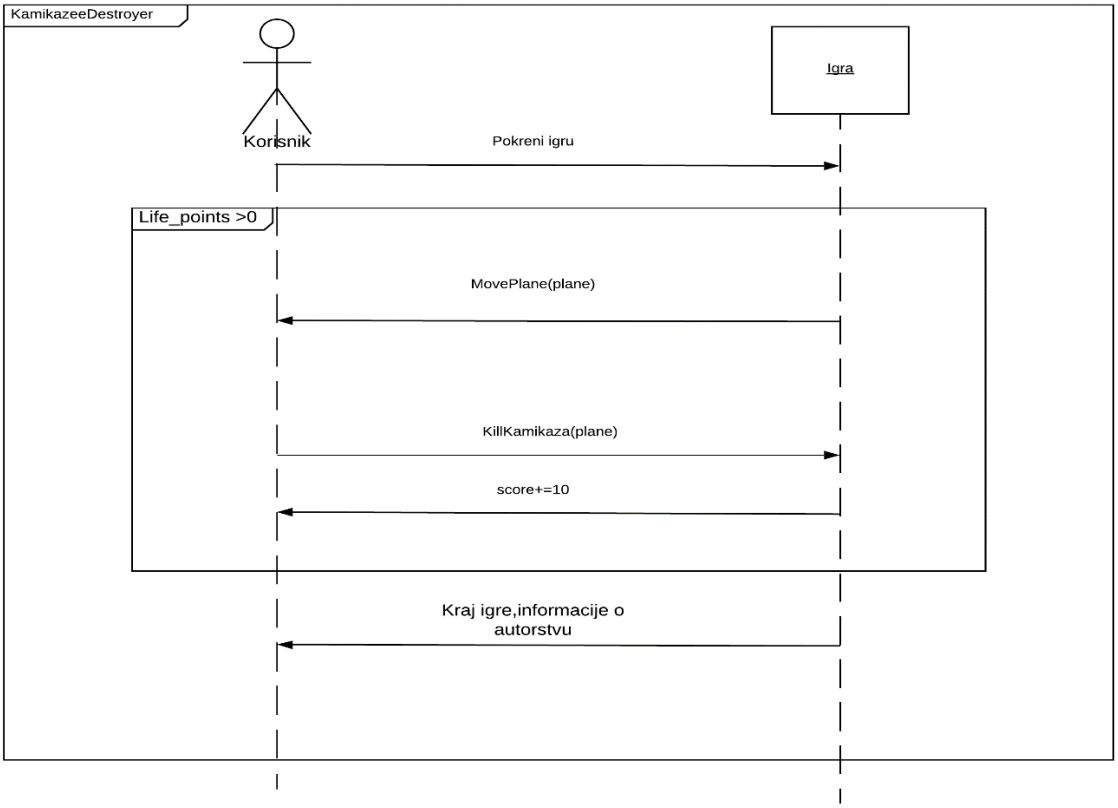
1. Ulaz i učitavanje objekata
2. Uništavanje protivničkih objekata + povećanje rezultata
3. Snižavanje igračevih 'životnih bodova'
4. Zapis o autorstvu i zahvala

Pošto se aplikacija sastoji od klasa i elementi iz svake klase ,bila ona parcijalna kao igrač ili zasebna kao klasa za 'bullet', svaka od njih ulazi u klasu za upravljanje igrom 'game'. Unutar klasnog dijagrama vidljivi su atributi svake klase, njihove funkcije i parametri koji ulaze u tu funkciju. U kasnijoj obradi igre objašnjeno je kroz programski kod zbog čega su klase tako napravljene. Dijagrami vidljivi na slikama 3,4 i 5.



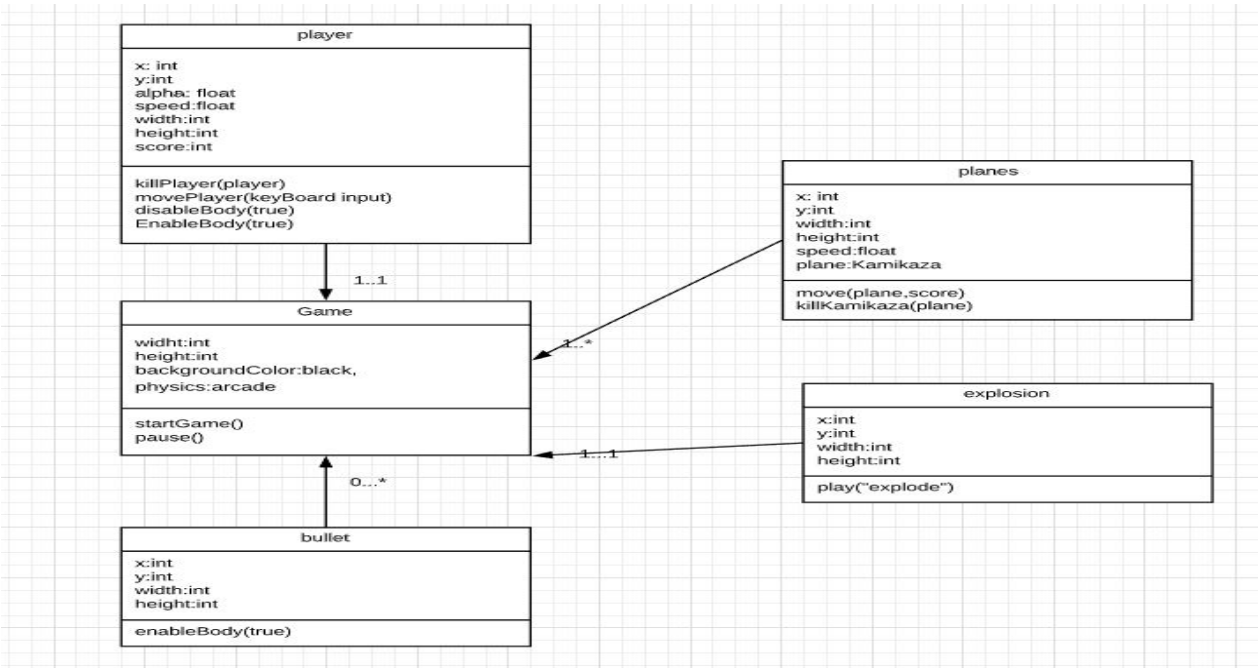
Slika 3. Use case dijagram,

Izvor:Vlastiti rad



Slika 4. Sequence dijagram,

Izvor:Vlastiti rad



Slika 5. Klasni dijagram,

Izvor:Vlastiti rad

Igra se sastoji od datoteka :

1.Index.html- Uključene sve dolje navedene datoteke unutar 'heada', 'script ima tip 'javascript', dok je izvor 'game.js' tj. konfiguracija ekrana igre.

2. Phaser.min.js.- Biblioteka s funkcija pomoću koje je ova igra izgrađena.
3. Explosion.js - klasa koja uključuje animaciju eksplozije unutar klase 'scene2.js'
4. Bullet.js- klasa koja uključuje izgled i površinu objekta 'bullet'
5. Scene1.js – klasa za učitavanje audio-vizualnih objekata
6. Scene2.js – klasa sa svim interakcijama između objekata

Način povezivanja unutar 'index.html' prikazano je na slici 6.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Phaser Game</title>
        <script type="text/javascript" src="phaser.min.js"></script>
        <script type="text/javascript" src="explosion.js"></script>
        <script type="text/javascript" src="bullet.js"></script>
        <script type="text/javascript" src="scene1.js"></script>
        <script type="text/javascript" src="scene2.js"></script>


        <script type="text/javascript" src="game.js"></script>
    </head>
    <body>
    </body>
</html>
```

Slika 6. index.html-kod,

Izvor:vlastiti rad

3.1 Ulazni zaslon

Ulazak u igricu se prikazuje kao dugme koje moramo s lijevim klikom miša pritisnuti kako bi pokrenuli igranje. Datoteka se zove 'scene1.js' i u njoj se vrši učitavanje svih objekata(zvučnih i vizualnih) i inicijalizacija dugmeta koji pokreće funkciju 'StartGame' s pozivom druge klasa i njenog konstruktora. Na taj način se pokreće datoteka 'scene2.js' s kojom počinje naše igranje. Objekti koji se učitavaju su pozadina 'more', naš igrač 'player', protivnici 'kamikaza','kamikaza1' i 'kamikaza2', efekt tj. animacija eksplozije 'explosion' , projektil 'bullet' i završni zaslon 'kraj'. Svaki od njih je predstavljen kao 'spritesheet' tj. kao slika u više okvira, iako neki od njih to nisu. Razlog tomu je što se ne uklapaju dobro s animacijom eksplozije ukoliko su stavljeni kao slika iz jednog okvira. Što znači kada dođe do eksplozije oni nestanu na određenoj točci,a za njima dođe eksplozije koja izgleda kao je proizašla iz površine,a ne iz objekta. Učitavanje slika i zvuka vidljvo na slici 7.

```
preload(){
  this.load.image("start","assets/button_start-game.png",{
    frameWidth: 200,
    frameHeight:250,
  });

  this.load.image("more","assets/more.jpg");
  this.load.image("kraj","assets/kraj.png");
  this.load.image("kamikaza1","assets/kamikaza1.png",{
    frameWidth: 150,
    frameHeight: 150,
  });
  this.load.spritesheet("kamikaza2","assets/kamikaza2.png",{
    frameWidth: 80,
    frameHeight: 80,
  });
  this.load.spritesheet("kamikaza3","assets/kamikaza3.png",{
    frameWidth: 80,
    frameHeight: 80,
  });
  this.load.spritesheet("player","assets/player.png",{
    frameWidth: 75,
    frameHeight: 75
  });
  this.load.spritesheet("explosion", "assets/explosion.png",{
    frameWidth: 77,
    frameHeight: 73
  });
  this.load.spritesheet("bullet","assets/bullet.png",{
    frameWidth: 60,
    frameHeight: 70

  });
  this.load.bitmapFont("pixelFont", "assets/font.png", "assets/font.xml");
  this.load.audio("pozadinska","assets/pozadinska.mp3");
  this.load.audio("explozija","assets/explozija.mp3");
}
```

Slika 7. Učitavanje izgleda objekata

Izvor:Vlastiti rad

Od nekih objekata možemo izraditi i animaciju. Npr. od našeg igrača smo izradili animaciju koja prikazuje 'mlaz goriva', dok smo od objekta 'explosion' napravili animaciju eksplozije. To je bilo moguće zbog toga što Phaser posjeduje u svojoj biblioteci atribut 'frameRate' koji označava koliko okvira se mijenja po sekundi. S atributom 'key' možemo pozvati animaciju, atributom 'repeat' dali se animacija ponavlja, ukoliko je -1 vrijednost atributa to znači da se ponavlja beskonačno mnogo, 0 znači samo jednom po interakciji,a ostali brojevi označavaju koliko puta se ponovi(slika 8).

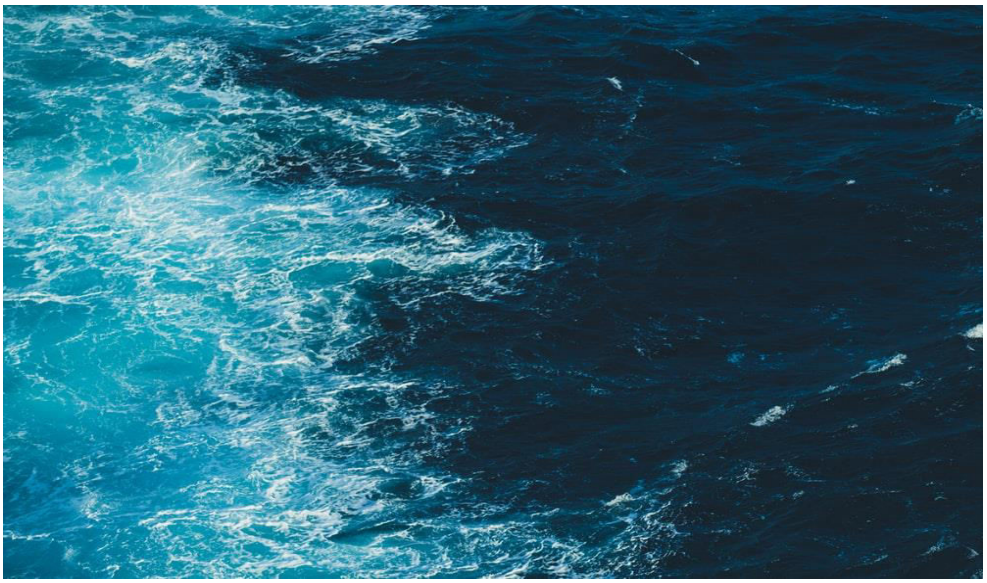
```
create() {
 this.start= this.add.image(400,300,"start");
 this.start.setInteractive();
 this.input.on('gameobjectdown',this.startGame,this);


  this.anims.create({
    key: "plane_anim",
    frames: this.anims.generateFrameNumbers("kamikaza"),
    frameRate: 1,
    repeat: -1
  });
  this.anims.create({
    key: "player_anim",
    frames: this.anims.generateFrameNumbers("player"),
    frameRate: 10,
    repeat: -1
  });


  this.anims.create({
    key: "explode",
    frames: this.anims.generateFrameNumbers("explosion"),
    frameRate: 30,
    repeat: 0,
    hideOnComplete: true
  });


 }
startGame(){
  this.scene.start("playGame");

}
```

Slika 8. Kreiranja animacije i funkcija za početak igre

Izvor:Vlastiti rad

3.2 Pozadina

Slika je preuzeta s izvora navedenog ispod slike9 i zaokrenuta za 90 stupnjeva te kao takva spremljena pod nazivom more.jpg. Igra započinje s učitavanjem pozadine.Ona se učitava na način da prvo odredimo koliko će biti velika, kod učitanja svih objekata u slici prije vidjeli smo da je ona 800x600,što zauzima cijeli prostor prikaza igre.Nakon toga u datoteci 'scene2' ona je započeta od koordinata (0,0) pošto ne smije prelaziti izvan prikaza ili biti skraćena za jedan dio. Kao i svi objekti i izgled pozadine i njezina konfiguracije veličine u koordinatnom sustavu se nalazi u datoteci 'preload.js'. Kreće se po y osi od manjem prema većem što znači da slika ide od gornjeg djela ekrana prema donjem(vidi pod sliku 10. i 11.)



Slika 9.Pozadina [3]

```
create(){

  this.background = this.add.tileSprite(0, 0, config.width, config.height, "more");
  this.background.setOrigin(0,0);
```

```
this.background.tilePositionY -= 2;
```

Slika 10 i 11. Podešavanje veličine slike i kretanja,

Izvor:Vlastiti rad

3.3 Igrač

Karakter s kojim upravlja igrač učitan je kao 'spritesheet' što znači da posjeduje više okvira unutar slike. U uvodu je spomenuto mogućnosti kretanja igrača koja su implementirana kao i na slici ispod. Bilo nam je olakšano jer 'phaser 3' knjižnica funkcija prepoznaje pod 'cursorKeys' atribute poput 'cursorKeys.right' za desnu strelicu itd,'cursorKeys.up' za kretanje prema dolje(slika 12). Također je moguće kretanje dijagonalno ukoliko igrač pristine dvije tipke suprotnog smjera tj. gore-desno ili dolje-lijevo itd. Na sljedećoj slici je prikazan izgled našeg karaktera te njegovo mijenjanje izgleda kroz 'spritesheetove'(slika 13). Postavljen je i 'rate' tj. brzina mijenjanja okvira po sekundi , tako da imamo dojam kako on zapravo koristi gorivo.

```
movePlayer(){
  if(Phaser.Input.Keyboard.JustDown(this.spacebar)){
    this.shootBullet();
  }
  this.player.setVelocity(0);
  if(this.cursorKeys.left.isDown){

    this.player.setVelocityX(-gameSettings.playerSpeed);
  }
  else if(this.cursorKeys.right.isDown)
    {

      this.player.setVelocityX(gameSettings.playerSpeed);


    }

    if(this.cursorKeys.up.isDown)
    {

      this.player.setVelocityY(-gameSettings.playerSpeed);

    }
    else if(this.cursorKeys.down.isDown)
    {

      this.player.setVelocityY(gameSettings.playerSpeed);

    }
  }
```

Slika 12. Postavke kretanja igrača

Izvor:Vlastiti rad

Slika 13. Naš igrač kroz okvire[4]

3.4 Protivnici

U ovoj igri smo protivnike nazvali 'kamikaze' zbog toga što se obrušavaju na našeg igrača. Svatko od njih ima zasebnu putanju kretanja. Kao što je navedeno ,postoje 2 osi, x bi mogli predstaviti kao smjer kretanja (lijevo-desno) dok je y brzina(okomito, suprotne orijentacije od našeg igrača). Ukoliko protivnik izađe izvan koordinatnog sustava on će se resetirati po random varijabli na x osi i na 0 koordinati y osi(*slika 22*). Na taj način se dobije privid kako neprijatelja ima beskonačno mnogo. Isti postupak je i ukoliko protivnik bude uništen.Kako bi se protivnici nastavili kretati morali smo u posebnu funkciju pod nazivom 'update' postaviti pozive za funkcije kretanja. Funkcije za kretanje primaju 2 parametra : na kojeg protivnika se i koja mu je brzina. Brzina raste proporcionalno s poboljšanjem rezultata igrača(prikazano na slici 14) . Svaka funkcija ima jednadžbu za kretanje. Također posjeduje i 'slowdown' tj. varijablu koja će usporiti kretanje nakon što igrač prvi put pogriješi(implementacija vidljivina na slici 15). Igrač ima pravo 3 puta pogriješiti .

```
this.moveKamikaza3(this.kamikaza3,this.score);
this.moveKamikaza2(this.kamikaza2,this.score);
this.moveKamikaza1(this.kamikaza1,this.score);
```

Slika 14. Poziv funkcija za kretanje protivnika

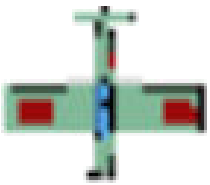
Izvor: Vlastit rad

```
moveKamikaza3(plane,score)
{
  if(plane.y>config.height)
  {
    this.resetPlane(plane);
  }
  plane.y+=2 + score/100 - this.slowdown;
  plane.x+=3.5;
  if(plane.x>config.width){
    this.resetPlane(plane)
  }
}
moveKamikaza1(plane,score)
{
  if(plane.y>config.height){
    this.resetPlane(plane);
  }
  plane.x+=+0.6;
    plane.y+=2 + score/120 - this.slowdown;
}
moveKamikaza2(plane,score)
{
  if(plane.y>config.height){
    this.resetPlane(plane);
  }
  plane.x-=1;
  plane.y+=2+score/130 - this.slowdown;
}
```

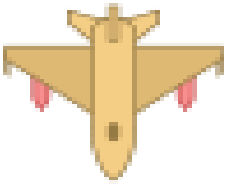
Slika 15. Funkcije za kretanje protivnika,

Izvor:Vlastiti rad

Vizualni izgled protivnika vidljivi su na slikama 16,17 i 18.



Slika 16. Kamikaza1-protivnik prvi [4]



Slika 17.Kamikaza2-protivnik drugi [4]

Slika 18. Kamikaza3-protivnik treci [4]

3.5 Posebna izdvojena klasa za objekt 'bullet'

Projektil se ispaljuje kada pritisnemo tipku 'space' .Prije toga njegova pozicija sa sakrivenim tijelom se nalazi ispod tijela objekta igrača i to pomaknuta od igrača za -2 po x osi te po -16 na osi y. Nakon što se pozove klasa i i postavlja objekt na klasi na kojoj smo trenutno tj. scene2 pošto samo u toj klasi postoji poziv. Nakon što je dodana u 'scene2' tijelo dobiva oblik i kreće se po -600 na y osi i postavlja se u grupu 'projectiles'. Također funkcija u klasi 'Bullet' ima update kako bi se kretanje nastavilo dok ne dođe do 0 na y osi.

Objekti vrše interakciju ukoliko se minimalno jedna točka njihove pozicije nalazi na istoj koordinati točke drugog objekta. Tako naprimjer kada projektil dođe do protivničkog objekta, i jedan i drugi se unište, a protivnički objekt se resetira. Tu dobivamo privid kako 'kamikaza' ima beskonačno mnogo, a to je napravljeno pošto je igra arkadnog modela.Također ukoliko dođe do sudara igrača i protivnika, oba se uništavaju, potom resetiraju, no to samo ukoliko igrač ima minimalno jedan od tri životna boda.Kod za klasu 'bullet.js' nalazi se na slici 19, dok izgled projektila na slici 20.

```
class Bullet extends Phaser.GameObjects.Sprite{
    constructor(scene){
        var x=scene.player.x - 2;
        var y=scene.player.y - 16;
        super(scene,x,y,"bullet");
        scene.add.existing(this);

        scene.physics.world.enableBody(this);
        this.body.velocity.y = - 600;

        scene.projectiles.add(this);

    }
    update(){

        if(this.y < 1 ){
          this.destroy();
        }
    }
}
```

Slika 19. Posebna klasa 'Bullet',

Izvor: Vlastiti rad

Slika 20. Bullet, [5]

'This.physics.add.overlap' funkcija prima 2 objekta u igri, poziv funkcije ukoliko dođe do stanja poklapanja dvaju točki, četvrti parametar govori da li je potrebna daljnja izmjena u fizici igre, a pokazivač 'this' se odnosi na trenutni događaj i objekte koji su u spektru događaja pošto su objekti stavljeni u grupe.Grupe nastaju zbog toga da ne moramo pisati svaki objekt zasebno jer bi to uvećalo i kompliciralo sam kod igre. Pozivi funkcija unutar koda vidljivi na slici 21.

```
this.physics.add.overlap(this.player,this.enemyPlanes,this.killPlayer,null,this);
this.physics.add.overlap(this.projectiles, this.enemyPlanes, this.killKamikaza, null, this);
```

Slika 21. Poziv funkcija za uništavanje protivnika ili igrača,

Izvor:vlastiti rad

3.6 Funkcija igrač-protivnik

Kada se jedan od protivničkih zrakoplova sudari s igračevim poziva se funkcija 'killPLayer' koja nema parametara. U njoj postoje više poziva drugih funkcija kao resetPlane i resetPlayer tj.oboje se resetiraju , igrač gubi jedan životni bod, dok protivnici kreću s nešto smanjenom brzinom kretanja. Također izgled igrača nestane na 2 sekunde potom se jednu sekundu promjeni u oblik nešto prozirniji nego uobičajeno(atribut alpha koji inače označava popunjenost izgleda objekta, stavlja se na 0.5). Kada se pojavi puni oblik igrača tada se on može ponovno kretati kamo želi. Protivnički zrakoplovi se uspore za vrijednost varijable 'slowdown' koja dobiva vrijednost = rezultat igrača/200.

Također poziva se još klasa 'Explosion.js' koja je slična kao i klasi 'bullet.js'. Ona poziva funkciju za animiranje eksplozije,usporedno s eksplozijom javlja se i njen

zvuk koji je već prije učitan kao audio objekt. Kada igrač bude pogođen,a nema preostalih bodova za daljnje igranje , javlja se posebna pozadina s njegovim završnim rezultatom. Implementacija prikazana na slici 22.

```
killPlayer(player){
  if(this.player.alpha==1){
  this.life-=1;
  }
  this.slowdown=this.score/200;
  this.lifeLabel.text= "Life points " + this.life;
  if(this.life<1){

    this.kraj = this.add.tileSprite(0,0, config.width, config.height, "kraj");
    this.kraj.tilePositionX -=2;
    this.kraj.setOrigin(0,0);
    this.konacni=this.score;
    this.scoreLabel = this.add.bitmapText(200,400 , "pixelFont", "YOUR FINAL SCORE: " + this.score , 50);
    this.credits1Label = this.add.bitmapText(10,450 , "pixelFont", "This game was made by Ivan Antunovic, student of 3rd year bacc. Inf" , 30);
    this.credits2Label = this.add.bitmapText(10,480 , "pixelFont", "Mentor:doc.dr.sc. Tihomir Orehovacki-FACULTY of Informatics, Pula" , 30);
    this.credits2Label = this.add.bitmapText(10,520 , "pixelFont", "Purpose of making : final thesis" , 30);
    player.disableBody(true, true);
    this.time.addEvent({
      delay: 2000,

      callback: this.resetPlayer,
      callbackScope: this,
      loop: false
    });
  }
  this.resetPlane(this.kamikaza1);
  this.resetPlane(this.kamikaza3);
  this.resetPlane(this.kamikaza2);
var explosion = new Explosion(this, player.x, player.y);
player.disableBody(true, true);

this.time.addEvent({
  delay: 1500,

  callback: this.resetPlayer,
  callbackScope: this,
  loop: false
});
```

Slika 22. Funkcija 'killPlayer'

Izvor:Vlastiti rad

3.7 Funkcija projektil-protivnik

Ova funkcija se poziva kod dodirivanje objekta projektila koji je proizašao iz klase Bullet i neprijateljskog zrakoplova. Nakon eksplozije projektil se uništava,pokreće se animacija eksplozije iz klase 'Explosion.js' ,a protivnik se

resetira(nacin resetiranja prikazan na slici 23). Također se pokreće i zvuk eksplozije ,koji je za razliku od vizualnog djela, rađen unutar klase u kojoj se odvija. Konfiguracija zvuka je podešena unutar funkcije 'killKamikaza' Sa svakim pozivom ove funkcije rezultatu igrača se pribraja 10 bodova. Rezultat se prikazuje i na ekranu,kao i životni bodovi igrača. To je omogućeno sa 'scorelabelom', tj. tekstualni prikaz preko površine koji se nalazi unutar ove funkcije. Dodan je string „SCORE“ + this.score, što označava da se stringu score pridružuje i vrijednost varijable 'score' koju pretvaramo u string kako bi se rezultat igrača mogao prikazati(slika 24).

```
resetPlane(plane){

  plane.y=3;
  var random=Phaser.Math.Between(0,config.width);
  plane.x=random;

}
```

Slika 23. Funkcija za restart protivničkog zrakoplova,

Izvor:Vlastiti rad

```
killKamikaza(projectile,kamikaza){
  var explosion= new Explosion(this,kamikaza.x,kamikaza.y);
  projectile.destroy();
  var explozijaConfig = {

    mute: false,
    volume:2,
    rate: 2,
    detune: 0,
    seek: 0,
    loop: false,
    delay: 0
  }
  this.explozija.play(explozijaConfig);
  this.resetPlane(kamikaza);
  this.score+=10;
  this.scoreLabel.text= "SCORE" + this.score;
}
```

Slika 24. Funkcija za uništavanje protivničkog zrakoplova,

Izvor:Vlastiti rad

4. Korisničke upute

Igrač ulazi u igru pritiskom dugmadi 'Start Game' . Nakon toga igra se pokreće,protivnici se kreću od vrha ekrana prema dolje, svaki sa svojom orijentacijom horizontalno po x osi i okomito po y osi.

Korisniku je nemoguće predvidjeti kamo će krenuti sljedeći zrakoplov stoga se jedino može osloniti na svoj refleks. S tipkom 'Spacebar' omogućeno je ispaljivanje projektila i uništavanje protivnika kako bi se dobio što veći rezultat, kao što je vidljivo na slici 25,a zapis o uspješnosti rezultata i životnih bodova na slici 27.

Svaki put kada igrač izgubi jedan od života, protivnici će naglo usporiti kao bi se igrač pripremio za novi pokušaj. Kada se igračev objekt ponovno stvara na dnu on ima onemogućeno kretanje na jednu sekundu,ali i dalje može ispaliti projektil kako bi se riješio potencijalne opasnosti i prebrzo izgubljenog novog boda. Takvo usporeno kretanje ne traje dugo jer se varijabla 'slowdown' zapravo povećava samo kada igrač izgubi jedan životni bod. Stoga se protivnici naglo ponovno akcelerirano ubrzavaju i igrač je ponovno na težinskoj razini kao prije prethodno izgubljenog životnog boda.Izgled igračevog objekta se također mijenja i polagano dolazi ponovno na pozadinu igre. Također postavljen je i vremenski 'delay' odnosno, odgoda interakcije s protivnicima dok se oblik igračevog objekta ne vrati na stari vizualni volumen.Uočljiva je promjena između slika 25. i 26.

Nakon što igrač izgubi sva tri životna boda, prikazuje mu se završna scena koja mu daje informacije o rezultatu i autorstvu igre. Ona je sastavljena od objekta koju sadrži sliku koja se kreće po x osi u gornjem djelu ekrana i 'labela' koji pružaju ostale informacije. Također zvuk je i dalje prisutan,a igra je pauzirana kako ne bi došlo do daljnjeg mijenjanja rezultata. Na ekranu je ispisana zahvala 'You served well' što u prijevodu znači „Odslužili ste dobro“.Ispod nje se nalazi zapis o autorstvu(slika 28).

Slika 25. Eksplozija protivničkog zrakoplova unutar igre,

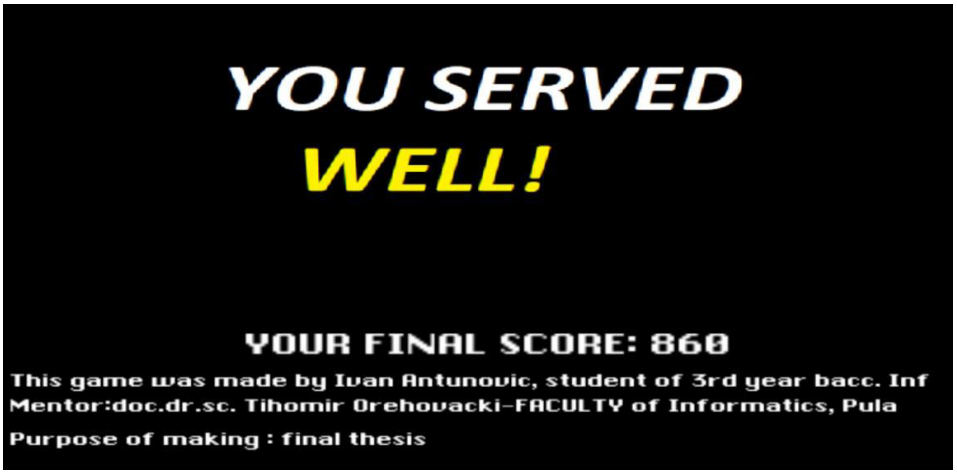
Izvor:Vlastiti rad

Slika 26. Igrač nakon sudara(dolje lijevo od sredine)

Izvor:Vlastiti rad

Slika 27. Prikaz rezultata i životnih bodova,

Izvor:Vlastiti rad



Slika 28. Posljednji prikaz u igri,

Izvor:Vlastiti rad

5. Zaključak

Cilj ove aplikacije bio je upoznavanje s jednim djelom JavaScript jezika i načina uključivanja drugih biblioteka i klasa koje se ne nalaze u istoj datoteci. Kako je igra u krajnosti zapravo prevedena pretraživaču u HTML jeziku može se uvidjeti da se u današnje vrijeme zaista mnogo toga može implementirati kroz različite jezike koji zajedno daju prvenstveno zanimljiv i koristan spoj. Također se može proučiti i shvatiti ostale pakete unutar okruženja 'node.js' kao što je 'Pixi.js' s kojim se mogu mnoge veće stvari napraviti pošto svakom objektu kojeg prenosi na server daje oblik.

Iako ova igra na prvi pogled nema neku veliku svrhu osim zabave, ovakva vrsta igara se zapravo koristi i u obučavanju vrhunskih vojnih pilota kako bi se unaprijedio refleks oka. Igra postoje sve kompliciranija i težina igranja ne ide dalje od razine reagiranja igrača. Tako po rezultatima se može postaviti prosječna vrijednost igrača ili na neki drugi način postaviti kriterij za prolaznost testova.

Greške i 'bugovi' u igrici su izbjegnuti točnim manjim matematičkim funkcijama i vremenskim odgodama. Također greške koje igrač može učiniti i prije igranja igre je da ne instalira 'node.js' ili da ne shvati koncept igre. No to je pokušano spriječiti s detaljnim opisom ispod dugmeta ' StartGame'. Također kao što je navedeno, igrač se može pribrati na manje vrijeme s usporavanjem protivnika. Igrač svakako mora primijetiti kada je pogriješio s animacijom eksplozije i ponašanjem njegovog objekta koji prvo postaje proziran, zatim mu se volumen vraća u normalu.

Također dizajn igre je jednostavan za krajnjeg korisnika i sve bitne informacije dostupne su mu u svakom trenutku preko samog sučelja. Glazba i zvukovi također poboljšavaju sam ugođaj igranja. Izrada ovakvih vrsta aplikacija doprinosi upoznavanju sa sve većom i brzorastućom industrijom igara i zabave u kojoj informatička tehnologija daju značajan doprinos . Čak

toliko da već duže vrijeme takva vrsta implementacije postaje zasebna grana u znanosti.

Literatura:

[1] https://onlinepngtools.com

[2] https://audiotrimmer.com

[3] https://it.depositphotos.com/98794384/stock-photo-blue-sea-texture-with-waves.html

[4] https://icons8.com/icons/set/airplane

[5] https://icons8.com/icons/set/bullet

Popis slika: